Decomposition of Unitary Matrices for Finding Quantum Circuits

Anmer Daskin[1] and Sabre Kais[2]

[1]*Department of Computer Science, Purdue University, West Lafayette, IN, 47907 USA*
[2]*Department of Chemistry and Birck Nanotechnology Center, Purdue University, West Lafayette, IN 47907 USA*

Constructing appropriate unitary matrix operators for new quantum algorithms and finding the minimum cost gate sequences for the implementation of these unitary operators is of fundamental importance in the field of quantum information and quantum computation. Evolution of quantum circuits faces two major challenges: complex and huge search space and the high costs of simulating quantum circuits on classical computers. Here, we use the group leaders optimization algorithm to decompose a given unitary matrix into a proper-minimum cost quantum gate sequence. Using this procedure, we present the circuit designs for the simulation of the Toffoli gate, the amplification step of the Grover search algorithm, the quantum Fourier transform, the sender part of the quantum teleportation and the unitary propagator of the Hamiltonian for the hydrogen molecule. The approach is general and can be applied to generate the sequence of quantum gates for larger molecular systems.

I. INTRODUCTION

Quantum computation promises to solve fundamental, yet otherwise intractable problems in many different fields. It is commonly believed that advancement in quantum computing science will bring new polynomial time algorithms, and some NP-complete problems shall be solvable in polynomial time, too. To advance the quantum computing field, finding circuit designs which can execute algorithms on quantum computers (in the circuit design model of quantum computing) is important. Therefore, it is of fundamental importance to develop new methods with which to overcome the difficulty in forming a unitary matrix describing the algorithm (or the part of the computation), and the difficulty to decompose this matrix into the known quantum gates [1]. Realizing the theoretical problems of quantum computers requires the overcoming decoherence problem [2]. Recently, West et al. demonstrate numerically that high fidelity quantum gates are possible in the frame work of quantum dynamic and decoupling [3].

The problem in the decomposition of a given unitary matrix into a sequence of quantum logic gates can be presented as an optimization problem. Williams and Gray [4] suggested the use of genetic programming technique to find new circuit designs for known algorithms, and also presented results for quantum teleportation. Yabuki, Iba [5] and Peng et al. [6] focused on circuit designs for the quantum teleportation by using different genetic algorithm techniques. Spector [7] explains the use of genetic programming to explore new quantum algorithms. Stadelhofer [8] used genetic algorithms to evolve black box quantum algorithms. There are also some other works [9–11] which evolve quantum algorithms or circuits by using genetic programming or genetic algorithms. Review of these procedures can be found in [12].

Evolution of quantum circuits faces two major challenges: complex and huge search space and the high costs of simulating quantum circuits on classical computers. Previous works focused on specific test cases and small scale problems. In this paper, we use our recently developed group leaders optimization algorithm (GLOA)[13]-an evolutionary algorithm-to decompose the unitary matrices representing some quantum algorithms or the unitary propagator of a given many-body Hamiltonian. Using two construction methods to find the unitary matrix representation of control gates, we have been able to efficiently use various kinds of quantum gates in the optimization process without increasing the computation time. In addition to explaining the details of the approach, we show the result circuit designs for the operators of the Grover search algorithm; the sender part of the quantum teleportation; the Toffoli gate; and the quantum Fourier transform (while some of the results are new circuit designs, some of them are known circuit designs). For the simulation of the hydrogen Hamiltonian, we give two circuit designs which have much less cost than the circuit design found by mapping the Hamiltonian from fermionic operators to the Pauli operators [14].

This paper is organized as follows: after defining the objective function to be minimized, we describe the algorithm used in the optimization; sec.IV are devoted the implementation details; and in Sec.V the result circuit designs are presented and discussed.

II. DEFINING OBJECTIVE FUNCTION

The definition of the objective function is an important factor in the optimization process. In our case, there are two factors which need to be optimized within the objective function: the correctness and the cost of the circuit. Let U_f be the matrix found by the optimization and U_g is the given unitary matrix to be decomposed. The correctness is defined in [15] as:

$$C = \left| \frac{Tr\left(U_g U_f^\dagger\right)}{N} \right|, \tag{1}$$

where C is the correctness; $N = 2^n$ (n is the number of qubits); the symbol † represents the complex conjugate transpose of a matrix; and Tr(..) is the trace of a matrix. This definition ignores the global phase differences which are physically indistinguishable; hence, makes the optimization easier by diversifying the reachable solutions for a problem in complex space. C obtained from Eq.(1) determines how much U_f looks like U_g and is in the range $[0, 1]$ since the product of two unitary matrices is another unitary matrix all eigenvalues of which have absolute value 1, and when $U_f = U_g$, $|Tr(U_f U_g^\dagger)|$ is equal to 2^n; hence, the value of the correctness is 1.

The cost of a circuit describes the level of ease with which this circuit is implemented; in order to make the implementation of a circuit easier and the circuits less error-prone, the cost of a circuit also needs to be optimized by minimizing the number of gates in the circuits. Hence, in addition to the correctness, the objective function should include the cost of the circuit.

However, defining the cost of a circuit is not an easy task due to the fact that each quantum computer model may have a different cost for a given quantum gate. Here, we simply take into consideration the number of qubits on which a gate is operating and the number of control qubits the gate has; these are the common factors increasing the implementation cost of a gate in all quantum computer models. Therefore, because quantum operations on distant qubits may not be feasible to implement physically [16], for the implementations of the circuits which requires the control of many qubits it may be possible to separate the circuit (the whole computation) into smaller parts for the partial implementation; in this sense, having quantum gates operating on closer or adjacent qubits in the design eases the implementation difficulty.

Considering the reasons in the previous paragraph, we define the cost of different types of quantum gates as follows: The cost of a single gate is 1 and less than the cost of any type of the control gates since the implementation of a single gate on quantum computers is much simpler than the implementation of the control gates. To estimate the cost of the control gate, instead of placing a constraint on the target and control qubits by requiring them to be adjacent as done in [16], we take the cost of a gate with the closer target and control qubits; the number of qubits between the control and target qubits is multiplied by a constant. This constant is 2 for the regular control gates and 3 for the multi-control gates. For instance, while the cost of the CNOT gate whose control and target qubits are neighbors is $2 \times 1 = 2$ (2 is the constant and 1 is the number of qubits between the target and control qubits.), the cost for the Toffoli gate is $3 \times 2 = 6$ (3 is the constant for the multi-control gates, and 2 is the number of qubits between the first control and the target qubits.). After finding the cost of each gate in a circuit, the cost of the whole circuit is defined as the sum up of the costs of the gates in the circuit. The following expression finds the cost of the circuit in Fig.1a:

$$Cost = 2 \times 1 + 2 \times 1 + 2 \times 2 + 2 \times 1 = 12. \quad (2)$$

The general objective function is defined by including both the correctness and the cost of the circuit with some weights:

$$y = \left| 1 - (\alpha C + \frac{\beta}{Cost}) \right|, \quad (3)$$

where the constants α and β are the weights for the correctness and the cost, and defined as:

$$\begin{aligned} 0 \leq (\alpha, \beta) \leq 1; \\ \alpha + \beta = 1. \end{aligned} \quad (4)$$

The cost in the objective function is also scaled to the range $[0, 1]$ by taking β less than 1. The expression αC must be dominant in the objective function in order to find exact or more accurate solutions. Because of the definition of the cost and the correctness, α should be much greater than β to make αC dominant. Therefore, the choices of α and β have effects on the number of iterations which is necessary to find a desired solution for a problem. Taking $\alpha = 0.9$ and $\beta = 0.1$ is the optimum choice which reduces the number of iteration and increases the correctness by making the correctness dominant in the objective function and giving higher priority to the correctness than the cost in the optimization process. Hence, in the experiments, we set $\alpha = 0.9$ and $\beta = 0.1$.

It is important to note that the value of the objective function never becomes zero since all circuit designs with at least one gate have a cost value. For the test cases, in addition to the values of objective functions, we give the C values for the circuits to make the cost and correctness values in the objective function more coherent. Also note that on the results of the objective function there may be some computer round-off errors.

III. GROUP LEADERS OPTIMIZATION ALGORITHM

Group leaders optimization algorithm (GLOA) described in more detail in [13] is a simple and effective global optimization algorithm that models the influence of leaders in social groups as an optimization tool. The general structure of the algorithm is made up by dividing the population into several disjunct groups each of which has its leader (the best member of the group) and members. The algorithm which is different from the earlier evolutionary algorithms and the pivot method algortihm [17–19] consists of two parts. In the first part, the member itself-the group leader with possible random part-and a new-created random solution are used to form a new member. If the formed new member is a better solution to the problem than the old member, it replaces the old one. This mutation is defined as:

$$\begin{aligned} new\ member = {} & r_1\ portion\ of\ old\ member \\ & \cup\ r_2\ portion\ of\ leader \\ & \cup\ r_3\ portion\ of\ random, \end{aligned} \quad (5)$$

where r_1, r_2, and r_3 determines the rates of the portions of the old member, the group leader, and the new-created random solution; which form the new member, and sum to 1.

In addition to the mutation, in each iteration for each group of the population one-way-crossover (also called the parameter transfer) is done between a chosen random member from the group and a random member from a different-random group. This operation is mainly replacing some random part of a member with the equivalent part of a random member from a different group. If new formed members give better solution to the problem, then they survive and replace the old members of the groups; otherwise, they do not. The amount of the transfer operation for each group is defined by a parameter called transfer rate. The values of the parameters of the algorithm used in the experiments are given in the next chapter.

The flow chart of the algorithm for our optimization problem is drawn in Fig.8. The more information about the algorithm can be found in ref.[13].

IV. IMPLEMENTATION DETAILS

A. Representation of Quantum Gates

In the optimization, we use similar representation method to the method of Cartesian Genetic Programming [20] in which each function set and inputs (in our case, gates and qubits) are represented as integers and genotypes including the inputs and the gates are represented as integer strings. The difference is: since the quantum gates may have an effect on the whole system, the gates should be represented in time steps; that means we cannot give the same inputs to two different gates at the same time as it is done in classical circuits. In the string of the genotypes each four numbers represent the gate, the qubit on which gate operates, the control qubit, and the angle for the rotation gates. The integers for the gates are determined by looking at the index of the gates in the gate set. For a gate set $\{V, V^{\dagger}, Y\}$, an example numeric string representing the circuit in Fig.1a is as:

2 3 2 0; **3** 2 1 0; **2** 3 1 0; **1** 3 2 0; **3** 2 1 0,

where the each group of four numbers separated by semicolons describes the each quantum gate in the circuit: the first numbers with bold fonts identify the gates, the last number is the angle, and the middle integers are the target and the control qubits, respectively, (the semicolons and the bold fonts do not appear in the real implementation).

B. Construction of the unitary matrices for the gates

Each quantum gate is considered as a unitary matrix which multiplies an input unitary matrix (in our cases an identity matrix): $U_i U_{input}$. Hence the sequence of U_i matrices represent the sequence of gates. For each evaluation of the objective function, a unitary matrix of order N for each gate in the circuit need to be constructed by taking the Kronecker product of the unitary matrix U_{Gate} and the required identity matrices. To find the matrix representation of different type of control gates, U_{Gate}, we use two similar pseudo codes which are run in $O(N)$ and $O(1)$ on a given identity matrix of order N and help to reduce the computation time of the optimization.

The main idea of these pseudo codes is: to use the order of qubits in the state register represented as $|q_0 q_1 ... q_n\rangle$ (n is the number of qubits on which the control gate acts); find the related indices for these qubits in a given identity matrix; and using the distance between the control and target qubits in the state register, to change the related elements of this matrix with the elements of the controlled elementary gate. The initial positions of the control and target qubits in the identity matrix are located by using the order of the control and target qubits in the state register, $|q_0 q_1 ... q_n\rangle$. And then, the jumping amount from this initial positions in the unitary matrix are determined by using the distance between the control and target qubits, and the elements in the related positions are changed according to the elements of the single elementary gate that acts on the target qubit, and is controlled by the control qubit.

1. *Pseudo code to construct the unitary matrix of a single-control quantum gate*

```
Input($target, control, U$)
{Either the control or the target qubit is zero.}
{U is a $2^{d+1}$ by $2^{d+1}$ identity matrix.}

1: $t = 2^{target}$;
2: $c = t + 2^{control}$;
3: $m = 2^{d-1} - 1$;
4: for $k = 0$ to $m$ do
5:   $i = c + 2k$;
6:   $j = t + 2k$;
7:   $U_{ii} = u_{00}; U_{ij} = u_{01}; U_{ji} = u_{10}; U_{jj} = u_{11}$;
   {where $u_{00}, u_{01}, u_{10}$, and $u_{11}$ are the
```

```
        matrix elements of the single gate.}
8:  end for
return $U$;
```

The above pseudo code builds a unitary matrix which represents the different types of the control gates possessing only one control and one target qubits. It locates the related elements of the identity matrix, and changes them with the elements of the elementary gate acting upon the target qubit.

In the code, d-which is the absolute difference between the control and target qubits-helps to find the number of qubits covered by the control gate and is the key element in the determination of the number of times the for-loop shall run. The matrix U is initially defined as a 2^{d+1} by 2^{d+1} identity matrix which shall be the representation of the control gate at the end of the algorithm. The variables c and t are the indices used to find the i and j which determine the indices of elements to be changed inside the loop. After finding the correct indices, the four elements of U are changed in each iteration with the elements (u_{00}, u_{01}, u_{10}, and u_{11}) of the single gate that operates on the target qubit. Since the loop runs $m + 1$ times, the total $4 \times (m + 1)$ elements are replaced by the elements of the single gate. At the end of the algorithm, the matrix U represents the desired unitary matrix for the control gate, and is given as an output of the algorithm.

The running time of this pseudo code is $O(N)$ in the worst case since the loop, which is the dominant in the running time, runs $m+1$ times.

2. Pseudo code to construct the unitary matrix of a multiple-control gate

```
Input($target, control, U$)
{Either the control or the target qubit is zero.}
{U is a $2^{d+1}$ by $2^{d+1}$ identity matrix.}

1:  $d = |control - target|$;
2:  $t = 2^{target}$;
3:  $c = t + 2^{control}$;
4:  $m = 2^{d-1} - 1$;
5:  $i = c + 2m$;
6:  $j = t + 2m$;
7:  $U_{ii} = u_{00}; U_{ij} = u_{01}; U_{ji} = u_{10}; U_{jj} = u_{11}$;
      {where $u_{00}, u_{01}, u_{10}$, and $u_{11}$ are the
      matrix elements of the single gate.}
return $U$;
```

Unlike in the regular-control gates, all of the qubits between the control and target qubits are also the control qubits in the multi-control gates. The idea of this pseudo code is almost the same as the previous pseudo code, but the absence of a for-loop makes this simpler than the first one and run in $O(1)$.

V. RESULTS AND DISCUSSIONS

In the experiments, some parameter adjustments have been done as follows: The default gate set consists of the rotation gates, the Pauli operators, the square root of not (V), the complex conjugate of V gate ($V^\dagger$), and the controlled type of these single gates. For the matrix representation of single quantum gates used in this paper, please refer to [1, 21, 22]. While in the case of the unitary propagator of the Hamiltonian of the hydrogen the angle is defined as a multiple of 0.005 and in the range $[0, 2\pi]$, in the other cases it is defined as $k\pi$, and k is multiple of 0.125 and in the range $[0, 2]$. During the evaluation of the objective function in the optimization, all quantum gates, except single gates, are formed by using the pseudo codes given in the previous section.

The parameters for the optimization algorithm are given in Table I. The correctness (C) and the minimized objective function values and the number of iterations are noted in Table II with respect to all circuit design results that are given in the paper.

The circuit diagrams in figures are drawn by using the output of the optimization program. As an example, the output of the program for the decomposition of the unitary matrix for the diffusion part of the Grover search algorithm is as follows:

$$\begin{array}{cccc}
G & T & C & Q \\
\hline
\text{Control X,} & 2 & 1 & 0 \\
0 & 0 & 0 & 0 \\
\text{Single V} & 2 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\text{Single V} & 1 & 0 & 0 \\
\text{Control X} & 2 & 1 & 0 \\
\text{Single V} & 1 & 0 & 0 \\
0 & 0 & 0 & 0
\end{array}, \tag{6}$$

where the column G represents the name of the gate, T is the target qubit, C is the control qubit, and Q determines the angle values for rotation gates. The number of rows is equal to the maximum number of gates. The zeros are the control gates with the equal control and target qubits, or a field the gate does not need to use such as the control qubit field for the single gates. Fig.2a shows the circuit diagram for this output.

The test cases are divided into two categories: the known quantum algorithms (the details of these algorithms can be found in [1]) and the unitary propagator of the hydrogen Hamiltonian. In the former case, the result circuit design does not show much significant improvement over the known circuits; therefore some of them have the same length or the structure as the known circuit designs since a great deal of work has been done on these algorithms, and these cases except three qubit quantum Fourier transform can be considered as simple cases for the optimization because of the small size of the circuits which are formed by only non-rotation gates (the angles for rotation gates are also parameters to be found by the optimization algorithm; which makes the optimization procedure harder.). However, implementation difficulties of different circuit designs with the same cost and length may not be the same in different quantum computer models; hence, finding different circuits is important for the implementation. In addition, the results show the efficiency and the ability of the optimization procedure (see Table II). The result circuit designs for the quantum algorithms are shown in Fig.1, Fig.2, Fig.3, and Fig.4. The case of the hydrogen Hamiltonian which finds the low cost circuit designs is explained in detail in the following subsection.

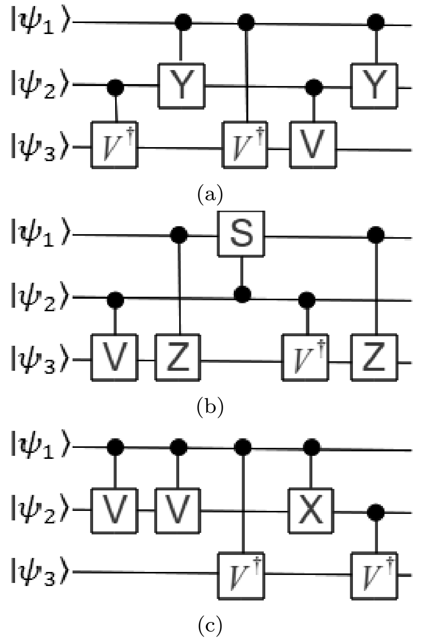

FIG. 1: The found circuit designs (a), (b), and (c) for the Toffoli gate.

A. The Hamiltonian of the hydrogen molecule

It has been shown that the ground and excited state energies of small molecules can be carried out on a quantum computer simulator using a recursive phase-estimation algorithm [23–25]. Lanyon et al. reported the application of photonic quantum computer technology to calculate properties of the hydrogen molecule in a minimal basis [14]. Here, we show how our method can be used to reduce the number of gates needed to perform the simulations.

Fermion model of quantum computation is defined through the spinless fermionic annihilation (a_j) and creation ($a_j^\dagger$) operators for each qubit j (j=1, ... , n), where the algebra of 2n

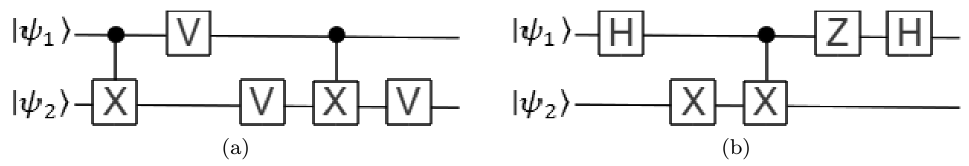
FIG. 2: The found circuit designs (a) and (b) for the two-qubit amplification part of Grover search algorithm.

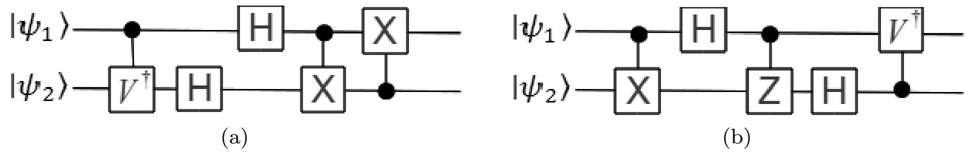
FIG. 3: The found circuit designs (a) and (b) for the two-qubit quantum Fourier transform.

elements obey the fermionic anti-commutation rules [26]:

$$\{a_i, a_j\} = 0,\ \{a_i, a_j^\dagger\} = \delta_{ij}, \tag{7}$$

where $\{A, B\} = AB + BA$ defines the anti-commutator. Using the Jordan-Wigner transformation, the fermion operators is mapped to the standard quantum computation operators through the Pauli spin operators [26]:

$$a_j \to \left(\prod_{k=1}^{j-1} -\sigma_z^k\right)\sigma_-^j = (-1)^{j-1}\,\sigma_z^1\sigma_z^2....\sigma_z^{j-1}\sigma_-^j$$
$$a_j^\dagger \to \left(\prod_{k=1}^{j-1} -\sigma_z^k\right)\sigma_+^j = (-1)^{j-1}\,\sigma_z^1\sigma_z^2....\sigma_z^{j-1}\sigma_+^j. \tag{8}$$

Once the electronic Hamiltonian is defined in second quantized form, the state space can easily be mapped to qubits. The electronic Hamiltonian in second quantized form is described as [14, 27, 28]:

$$H = \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2}\sum_{pqrs} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s, \tag{9}$$

where the integrals h_{pq} and h_{pqrs} evaluated during the Hartree-Fock procedure are defined as follows:

$$h_{pq} = \int dx \chi_p^*(x)\left(-\frac{1}{2}\nabla^2 - \sum_\alpha \frac{Z_\alpha}{r_{\alpha x}}\right)\chi_q(x) \tag{10}$$

and

$$h_{pqrs} = \int dx_1 dx_2 \frac{\chi_p^*(x_1)\chi_q^*(x_2)\chi_r(x_2)\chi_s(x_1)}{r_{12}}, \tag{11}$$

where $r_{\alpha x}$ is the distance between the α^{th} nucleus and the electron, r_{12} is the distance beteen electrons, ∇^2 is the Laplacian of the electron spatial coordinates, and $\chi_p(x)$ is a selected single-particle basis. Whitfield et al.[27] considered H in Eq.(9) as $H^{(1)} + H^{(2)}$. Since $h_{pqrs} = h_{qprs}$, and so $h_{ijji} = h_{jiij} = -h_{ijij} = -h_{jiji}$, they noted the parts of the Hamiltonian as follows:

$$H^{(1)} = h_{11}a_1^\dagger a_1 + h_{22}a_2^\dagger a_2 + h_{33}a_3^\dagger a_3 + h_{44}a_4^\dagger a_4, \tag{12}$$

and

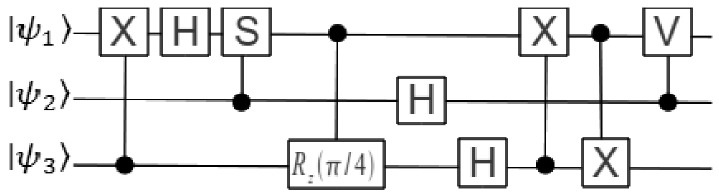

FIG. 4: The found circuit design for the three-qubit quantum Fourier transform.

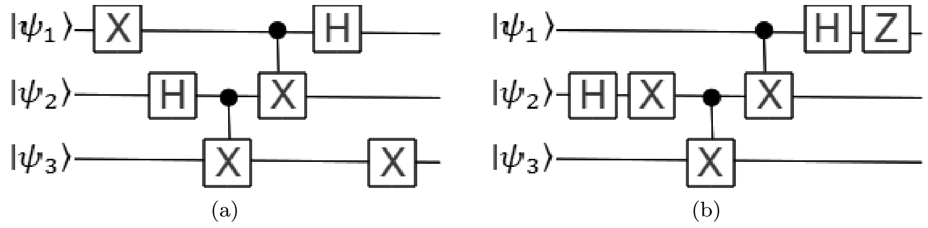

FIG. 5: The found circuit designs for the sender part of the quantum telportation.

$$\begin{aligned}H^{(2)} &= h_{1221}a_1^\dagger a_2^\dagger a_2 a_1 + h_{3443}a_3^\dagger a_4^\dagger a_4 a_3 + h_{1441}a_1^\dagger a_4^\dagger a_4 a_1 + h_{2332}a_2^\dagger a_3^\dagger a_3 a_2 \\ &+ (h_{1331} - h_{1313})a_1^\dagger a_3^\dagger a_3 a_1 + (h_{2442} - h_{2424})a_2^\dagger a_4^\dagger a_4 a_2 \\ &+ Re(h_{1423})(a_1^\dagger a_4^\dagger a_2 a_3 + a_3^\dagger a_2^\dagger a_4 a_1) + Re(h_{1243})(a_1^\dagger a_2^\dagger a_4 a_3 + a_3^\dagger a_4^\dagger a_2 a_1) \\ &+ Im(h_{1423})(a_1^\dagger a_4^\dagger a_2 a_3 + a_3^\dagger a_2^\dagger a_4 a_1) + Im(h_{1243})(a_1^\dagger a_2^\dagger a_4 a_3 + a_3^\dagger a_4^\dagger a_2 a_1).\end{aligned} \tag{13}$$

Using their findings [27] for the spatial integral values for atomic distance $1.401 a.u.$, the Hamiltonian and the unitary propagator for this Hamiltonian, e^{-iHt} (t was taken as 1), are found as matrices of order 16. The algorithm is run with the parameter values in Table I and Table II for the unitary propagator, and the decomposed circuit design is shown in Fig.6. It is important to note that the circuit in Fig.6 does not include the approximation error coming from the Trotter-Suzuki decomposition; however, it has some small errors which can be gaged from the values of the correctness and the objective function in Table II.

In addition to the exact unitary propagator, the unitary operator at bond distance 1.401 atomic units is found by simulating the approximated-general circuit designs for the simulation of the Hamiltonian in the paper [27] (In [27], the non-commuting terms in the Hamiltonian are approximated by following the Trotter-Suzuki decomposition, and the given approximated circuit design is a general circuit for the unitary propagator; the values of the angles in the quantum gates depend on the spatial integral values.). The decomposition of this unitary propagator is found as in Fig.7 with the value of correctness 0.9961. The circuit design in Fig.7 consists of 7 quantum gates while the complete circuit design in [27] consists of 87 quantum gates.

VI. CONCLUSION

To be able to simulate Hamiltonians of atomic and molecular systems and also apply quantum algorithms to solve different kinds of problems on quantum computers, it is necessary to find implementable quantum circuit designs including the minimum cost and number of quantum gate sequences. Since deterministic-efficient quantum circuit design methodology is an open problem, we applied stochastic evolutionary optimization algorithm, GLOA, to search quantum circuit designs for given unitary matrices representing algorithms or the unitary propagator of a molecular Hamiltonian. In this paper, in addition to explaining the ways of the implementation and design of the optimization problem, we give circuit designs for the Grover search algorithm, the Toffoli gate, the quantum Fourier transform, and the quantum teleportation. Moreover, we give the two circuit designs for the simulation of the Hamiltonian of the hydrogen molecule by decomposing the unitary matrix operators found by following the fermionic model of quantum computation and simulating the circuits given in [27]. In the case of the hydrogen molecule we reduced significantly the number of gates needed to simulate the unitary operator from 87 quantum gates to 7 quantum gates. The approach is general and can be applied to generate the sequence of quantum gates for larger molecular systems.

VII. ACKNOWLEDGMENTS

We would like to thank the Army Research Office (ARO) and the NSF Center for Quantum Information and Computation for Chemistry, award number CHE-1037992, for financial support of this project.

BIBLIOGRAPHY

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2000).

[2] D. A. Lidar, I. L. Chuang, and K. B. Whaley, Phys. Rev. Lett. **81**, 2594 (Sep 1998).

[3] J. R. West, D. A. Lidar, B. H. Fong, M. F. Gyure, X. Peng, and D. Suter(2009), arXiv:0911.2398v1.

[4] C. P. Williams and A. G. Gray, in *Quantum Computing and Quantum Communications* (Springer Berlin / Heidelberg, 1999) pp. 113–125.

[5] T. Yabuki and H. Iba, in *In Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference* (Morgan Kauffman Publishers, 2000) pp. 421–425.

[6] F. Peng, G. jun Xie, and T. hao Wu, International Conference on Convergence Information Technology **0**, 70 (2009).

[7] L. Spector, *Automatic Quantum Computer Programming: A Genetic Programming Approach (Genetic Programming)* (Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006).

[8] R. Stadelhofer, W. Banzhaf, and D. Suter, AI EDAM **22**, 285 (2008).

[9] A. Leier, *Evolution of Quantum Algorithms using Genetic Programming*, Ph.D. thesis, Dortmund University, Germany (jul # 21 2004).

[10] M. Lukac and M. Perkowski, in *EH '02: Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware (EH'02)* (IEEE Computer Society, Washington, DC, USA, 2002) p. 177.

[11] P. Massey, J. A. Clark, and S. Stepney, in *GECCO (2)* (2004) pp. 569–580.

[12] A. Gepp and P. Stocks, Genetic Programming and Evolvable Machines **10**, 181 (2009).

[13] A. Daskin and S. Kais(2010), arXiv/1004.2242.

[14] B. P. Lanyon, J. D. Whitfield, G. G. Gillett, M. E. Goggin, M. P. Almeida, I. Kassal, J. D. Biamonte, M. Mohseni, B. J. Powell, M. Barbieri, A. Aspuru-Guzik, and A. G. White, Nature Chemistry **2**, 106 (January 2010).

[15] T. Reid, *On the evolutionary design of quantum circuits*, Master's thesis, Waterloo University, Ontario, Canada (2005).

[16] S. Ding, Z. Jin, and Q. Yang, Soft Comput. **12**, 1059 (2008).

[17] P. Serra, A. F. Stanton, and S. Kais, Phys. Rev. E **55**, 1162 (Jan 1997).

[18] P. Nigra and S. Kais, Chemical Physics Letters **305**, 433 (1999).

[19] P. Serra, A. F. Stanton, S. Kais, and R. E. Bleil, The Journal of Chemical Physics **106**, 7170 (1997).

[20] J. F. Miller and S. L. Harding, in *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation* (ACM, New York, NY, USA, 2008) pp. 2701–2726.

[21] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator,

J. A. Smolin, and H. Weinfurter, Phys. Rev. A **52**, 3457 (Nov 1995).

[22] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing* (Oxford University Press, Inc., New York, NY, USA, 2007).

[23] A. Aspuru-Guzik, A. D. Dutoi, P. J. Love, and M. Head-Gordon, Science **309**, 1704 (2005).

[24] H. Wang, S. Kais, A. Aspuru-Guzik, and M. R. Hoffmann, Phys. Chem. Chem. Phys. **10**, 5388 (September 2008).

[25] D. A. Lidar and H. Wang, Phys. Rev. E **59**, 2429 (Feb 1999).

[26] G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme, Physical Review A **64**, 022319+ (Jul 2001).

[27] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik, *Quantum Computing Resource Estimate of Molecular Energy Simulation*, Tech. Rep. arXiv:1001.3855 (2010).

[28] E. Ovrum and M. Hjorth-Jensen, *Quantum computation algorithm for many-body studies*, Tech. Rep. arXiv:0705.1928 (2007).

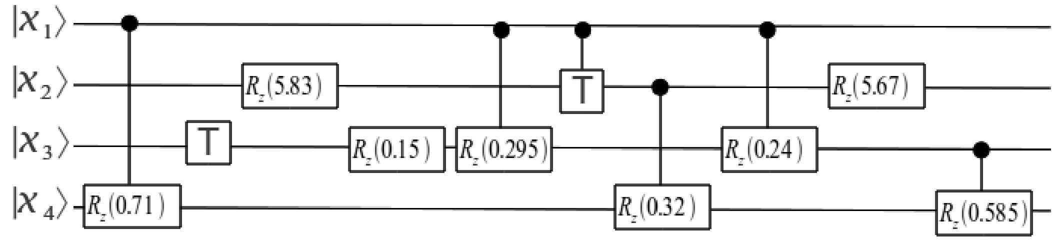

FIG. 6: The circuit design for the unitary propagator of the Hamiltonian of hydrogen molecule. The unitary propagator is found by using the spatial integral values in [27] and the definitions for the annihilation and creation operators in Eq.(8) into the Eq.(9), Eq.(12), and Eq.(13).

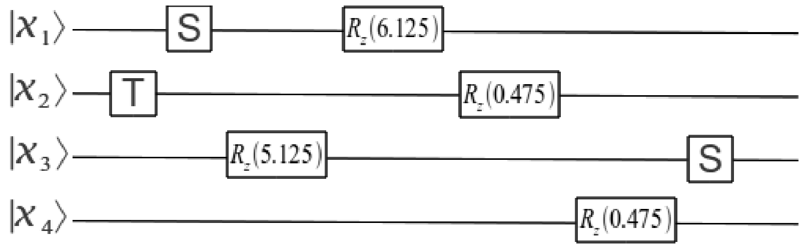

FIG. 7: The circuit design for the unitary propagator of the hydrogen Hamiltonian. The unitary propagator is found by simulating the circuits and pseudo-code given in [27].

TABLE I: The values of the parameters within the group leaders optimization algorithm

Parameter name	Number of groups	Number of population in each group	r_1	r_2	r_3	Number of transfer for each group
Value	15	25	0.8	0.1	0.1	$\frac{\text{Number of Variables}^{\dagger}}{2} - 1$

†Since each gate has four variables, the total number of variables is equal to four times maximum number of gates.

TABLE II: Optimization parameters and results for the test problems

The unitary matrix	Number of qubits	Maximum number of gates	Number of iteration	Objective function value	Correctness (C)
Toffoli Gate, Fig.1a	3	8	500	0.09167	1
Toffoli Gate, Fig.1b	3	8	500	0.09286	1
Toffoli Gate, Fig.1c	3	8	500	0.09286	1
Sender part of teleportation, Fig.5a	3	8	2500	0.08752	1
Sender part of teleportation, Fig.5b	3	8	2500	0.08752	1
Diffusion Operator of GSA, Fig.2a	2	8	500	0.08335	1
Diffusion Operator of GS, Fig.2b	2	8	500	0.08571	1
Two-qubit QFT, Fig.3a	2	8	500	0.08752	1
Two-qubit QFT, Fig.3b	2	8	500	0.08752	1
Three-qubit QFT, Fig.4	3	12	2500	0.09565	1
Simulation of the H_2 Hamiltonian, Fig.6	4	12	10000	0.1081	0.9870
Simulation of the H_2 Hamiltonian, Fig.7	4	12	10000	0.0892	0.9961

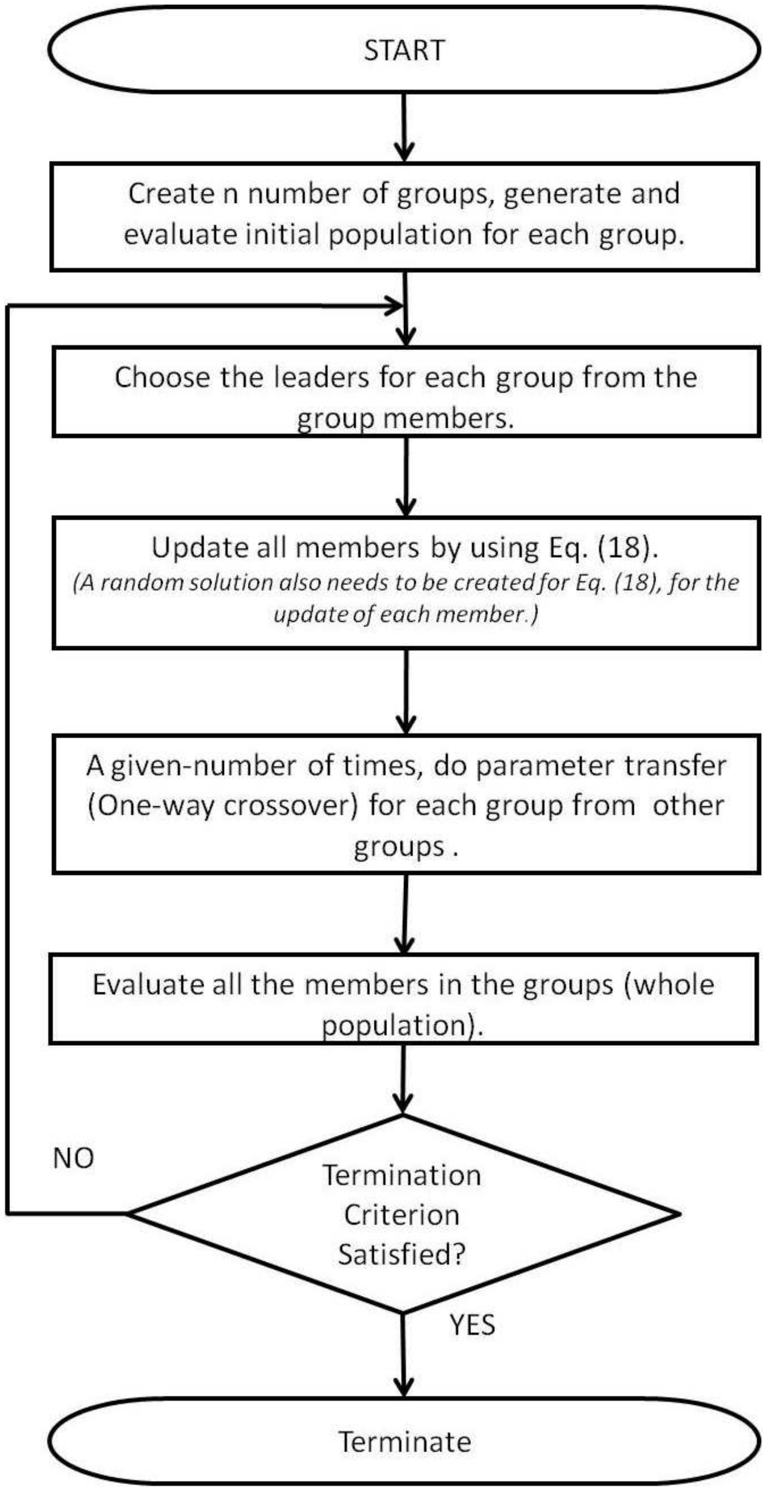
START
Create n number of groups, generate and evaluate initial population for each group.
Choose the leaders for each group from the group members.
Update all members by using Eq. (18).
(A random solution also needs to be created for Eq. (18), for the update of each member.)
A given-number of times, do parameter transfer (One-way crossover) for each group from other groups .
Evaluate all the members in the groups (whole population).
NO
Termination Criterion Satisfied?
YES
Terminate

FIG. 8: The flow chart of the group leaders optimization algorithm